Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Julijan Strajnar

Vizualizacija izbrane skladbe s pomočjo digitalne animacije

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Narvika Bovcon

Ljubljana, 2017

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi predstavite postopek izdelave glasbene vizualizacije od idejne zasnove do upodabljanja končnega izdelka. Opišite, kako ste izdelali vizualizacijo v izbranem programu za 3D modeliranje in animiranje ter kako ste v video vključili spremembe na podlagi glasbe.

Kazalo

Povzetek

Naslov: Vizualizacija izbrane skladbe s pomočjo digitalne animacije

Avtor: Julijan Strajnar

Z diplomsko nalogo sem želel ustvariti glasbeno vizualizacijo skladbe s podobami, ki sem si jih ustvaril, ko sem poslušal skladbo Singularity avtorja Stephana Bodzina. Odločil sem se za izdelavo predupodobljenega videa, ki sliko gradi s pomočjo digitalne animacije. Tekom diplomske naloge predstavim računalniške postopke za avtomatsko transkripcijo glasbe in izgradnjo vizualnega modela ter predstavim kriterije, na podlagi katerih sem izbral 3D-program, ki je bil za tip animacije in končni videz mojega izdelka najprimernejši. Predstavim tudi orodje, v katerem sem naredil animacijo, Cinema4D in njegove glavne vmesnike. Na koncu se podrobneje posvetim predstavitvi izdelave mojega diplomskega izdelka.

Ključne besede: digitalna animacija, avdio vizualizacija, 3D-animacija.

Abstract

Title: Visualization of a musical composition using digital animation

Author: Julijan Strajnar

For my diploma work, I wanted to create a musical visualization of the idea that I came up with when listening to Singularity by Stephan Bodzin. I decided to create a pre-rendered video that is made using digital animation. In my graduation thesis, I present computer procedures for automatic transcription of music, the construction of a visual model, and the criteria on which I chose the 3D-program that was the most suitable for the type of animation I wanted to make. I also introduce the tool in which I made the animation, Cinema4D and its main interfaces. In the end, I devote more detail to the production of my diploma animation.

Keywords: digital animation, audio visualization, 3D-animation.

Poglavje 1

Uvod

Vizualizacija je širok pojem, Slovar slovenskega knjižnega jezika opredeli pomen te besede kot: "vizualno predstavitev, upodobitev" nečesa, lahko besedila, glasbe ali česar koli drugega. Način upodobitve ni opredeljen, torej so možne različne upodobitve z različnimi metodami in tehnikami. Ključno je, da nečemu, kar samo po sebi nima vidne podobe (kot npr. zvok ali pa pomen besed) oz. vizualne predstavitve, tako podobo oz. predstavitev izdelamo. Gre torej za prevod med različnimi tipi zaznave in razumevanja. V diplomski nalogi sem se lotil problema upodobitve izbrane skladbe.

Glasbena vizualizacija je velik del glasbene industrije in vizualne umetnosti. Že umetniki pred stoletji so poskušali upodobiti glasbo na vizualen način v obliki različnih slik, ki so bile neke vrste glasbene notacije (grafični zapisi zvoka, na podlagi katerih je bilo možno glasbo poustvariti, kakor deluje tudi notni zapis) ali pa so ilustrirale samo doživljanje ob poslušanju glasbe na narativen ali pa abstrakten način – v smeri abstrakcije in raziskovanj sinestetičnega[1] doživljanja barve in zvoka je delal npr. Vasilij Kandinski [26]. Opera, različne glasbene predstave, film, animirani film, videospot – so bili pred prihodom računalnikov načini za dodajanje vizualne plasti, ki je spremljala glasbo.

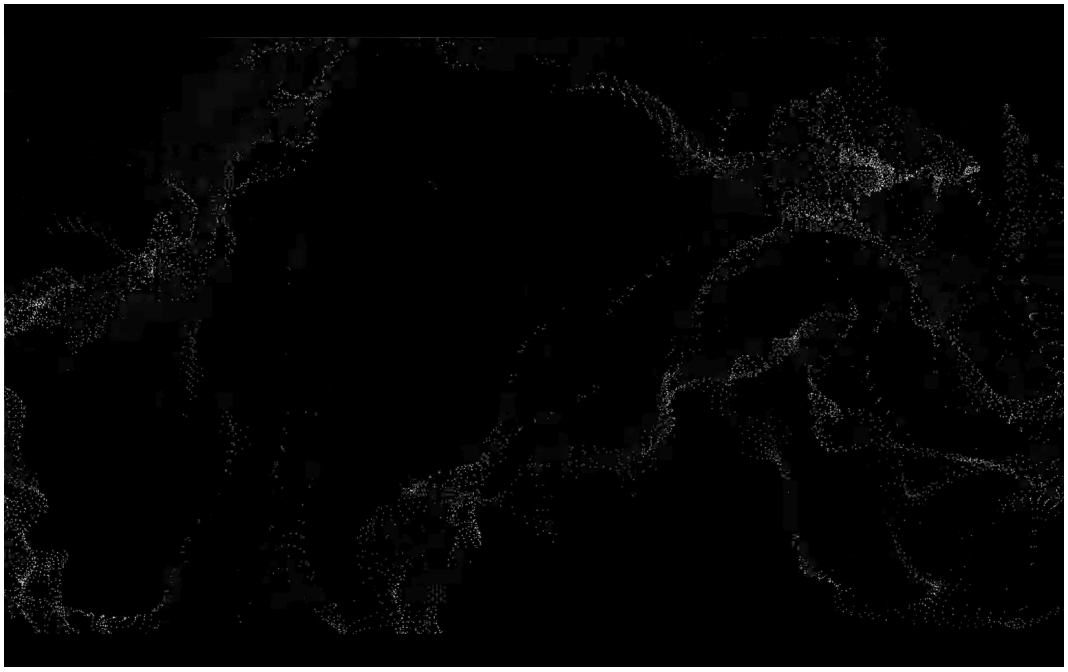
[1]Neposredni primer tega lahko vidimo pri ljudeh, ki imajo sinestezijo – nevrološko mešanje čutov. Sinesteti vidijo različne tone glasbe kot različne odtenke barve [31].

V zadnjih letih pa je računalniška tehnologija zabrisala mejo med glasbo in sliko in nam omogočila lažjo in neposredno vizualizacijo glasbe. Glasbena vizualizacija je lahko realnočasovna, ali pa ne. Primer realnočasovne vizualizacije glasbe je iTunes s svojim glasbenim vizualizatorjem. V 2. poglavju diplomske naloge predstavim računalniške postopke za avtomatsko transkripcijo glasbe in izgradnjo vizualnega modela, ki glasbi bolj ali manj ustreza [18]. Zaradi omejitev obeh pristopov sem se odločil, da se v diplomski nalogi raje kot avtomatični vizualizaciji glasbe posvetim izdelavi unikatne digitalne animacije. Ta bo namreč moje celostno doživljanje ob poslušanju izbrane skladbe upodobila bistveno bolj natančno: z več posebej za to skladbo izbranimi in natančno izdelanimi oblikami, ki nosijo pomen v odnosu do te skladbe, in z več detajli, ki so premišljeni in v okviru narativno-likovne celote nezamenljivi. Tega sem se v diplomskem delu lotil s svojo vizualno interpretacijo skladbe Singularity Stephana Bodzina.

Odločil sem se za izdelavo predupodobljenega videa, ki sliko gradi s pomočjo digitalne animacije, izdelane v izbranem 3D-programu. Računalniški program za izdelavo tridimenzionalnih modelov in njihovih upodobitev v slikah in animacijah bomo v diplomskem delu krajše imenovali z izrazom 3D-program, ki je v slovenskem jeziku uveljavljen izraz, saj je uporabljen v recenziranem univerzitetnem učbeniku za to področje [16]. V 3. poglavju diplomske naloge predstavim kriterije, na podlagi katerih sem izbral 3D-program, ki je bil za tip animacije in končni videz mojega izdelka najprimernejši in s pomočjo katerega sem bil pri delu lahko najbolj učinkovit. Digitalna animacija je namreč časovno zelo zahteven postopek, zato je ključnega pomena že na začetku izbor ustreznega orodja, ki nas pri delu ne bo oviralo. V 4. poglavju diplomske naloge predstavim izbrano orodje Cinema4D in njegove glavne vmesnike.

V 5. poglavju se podrobneje posvetim predstavitvi mojega diplomskega izdelka, dejanski izdelavi 7 minut in 18 sekund dolge digitalne animacije. Uradni videospot Bodzinove skladbe Singularity je bil s svojo temačno abstraktno podobo neposredna inspiracija za moje delo. Skladbo Singularity

sem razdelil na štiri dele, ki sem jih zaznal ob poslušanju skladbe kot celote. Za vsak del sem izdelal drugačno vizualno predstavitev, pri čemer sem inspiracijo črpal iz slik vesolja, mikroskopskega sveta in podmorskega sveta. Kako sem izdelal posamezne svetove, opišem natančno tudi s tehničnega vidika, saj pri digitalni animaciji podobe, dokler ni tehnično povsem izdelana, preprosto ni. Tako pri modeliranju, animiranju in izdelavi posameznih materialov sem naletel na izzive, ki sem jih moral rešiti z ustrezno uporabo različnih funkcionalnosti orodja Cinema4D. Poleg tehnične izdelave posameznih objektov v tem poglavju predstavim tudi okvir in korake delovnega postopka pri digitalni animaciji. Posebej poudarim iterativni pristop, v katerem izdelavi in upodabljanju posamezne prototipne rešitve sledi analiza podobe in gibanja, identifikacija napak in možnih popravkov ter ponovitev postopka z izboljšavo.



Slika 1.1: Primer glasbene vizualizacije iz uradnega videospota skladbe Singularity avtorja Stephana Bodzina[13].

Poglavje 2

Realnočasovna vizualizacija glasbe

Problem realnočasovne vizualizacije glasbe lahko razdelimo na dva podproblema. Prvi je avtomatska transkripcija digitalnih zvočnih posnetkov. S pomočjo pridobljene transkripcije lahko nato izgradimo vizualni model (to je drugi podproblem), kateri preslika zvočni prostor v vidni prostor.

2.1 Avtomatska transkripcija digitalnih zvočnih posnetkov

Transkripcija je postopek pretvorbe zvočnega posnetka v neko pisno obliko, kot naprimer zapis not. Pri vizualizaciji, ki sem jo imel v mislih, bi potreboval zelo natančno transkripcijo bobnov, šuma ter notni zapis sintetizatorja. Da lahko pridemo do takega zapisa, moramo najprej poznati osnovne lastnosti zvoka ter njegove predstavitve v digitalni obliki.

Zvok je mehansko valovanje, katerega zaznamo s pomočjo naših čutil. Glavne komponente signala so frekvence in amplitude. Amplituda nam pove, koliko je zvok glasen, frekvenca pa nam da informacijo o višini tona. Najbolj osnovna oblika zvoka je sinusno valovanje. Druga valovanja dobimo s pomočjo seštevanja sinusoidnega valovanja. Avdio signal večinoma vsebuje

frekvence od 20 do 20.000 Hz. To so frekvence zvoka, ki jih je človeško uho sposobno zaznati. Signal ima časovno in amplitudno os. Pri analognem signalu sta obe zvezni. Sinusni signal lahko opišemo s funkcijo:

$$x(t) = Asin(2\pi Ft + \theta) \tag{2.1}$$

A predstavlja amplitudo, t čas, F frekvenco, θ pa fazni kot. Ker potrebujemo za transkripcijo digitalno obliko skladbe, jo je treba vanjo najprej pretvoriti. To naredimo s pomočjo analogno-digitalnega pretvornika. Ta pretvornik vzorči analogni zvezni signal ter ga pretvori v diskretni signal. To naredi tako, da v določenem vzorčnem intervalu shrani vrednost signala. Torej dobimo signal, kateri je zvezen po amplitudi in diskreten po času. Sledi kvantizacija, katere rezultat je dodana diskretnost po amplitudi. Prišli smo do signala, ki je pripravljen za transkripcijo.

Nato signal razdelimo na več osnovnih sinusnih valovanj z različnimi amplitudami in frekvencami. Iz take oblike lahko lažje izluščimo transkripcijo, ki si jo želimo. Algoritem, ki dekompozira signal, se imenuje diskretna Fourierjeva transformacija (DFT). Poznamo pa še optimizirano verzijo - hitro Fourierjevo transformacijo (FFT). Hitra Fourierjeva transformacija se uporablja zato, ker ima časovno zahtevnost $O(nlogn)$, medtem ko ima DFT časovno zahtevnost $O(n^2)$.

DFT signala $X(n)$ je definiran s formulo

$$X_k = \sum_{n=0}^{N-1} x(n) \cdot e^{-2i\pi nk/N}, \qquad k = 0, 1, 2, \ldots, N-1. \tag{2.2}$$

Informacijo o amplitudi in fazi pa dobimo na podlagi naslednjih enačb:

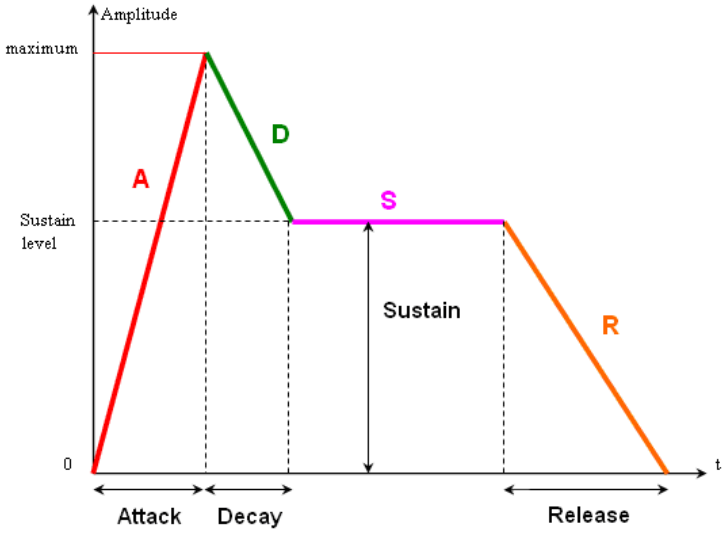
$$|X_k| = \sqrt{Re(X_k)^2 + Im(X_k)^2} \tag{2.3}$$

$$arg(X_k) = \arctan 2(Im(X_k), Re(X_k)), \tag{2.4}$$

Arctan2 je krožna funkcija arcus tangens z dvema argumentoma.

Za našo transkripcijo moramo izračunati noto oz. boben na določen časovni interval. Takrat pa pride v poštev malo spremenjena verzija prej

omenjenega algoritma, imenovana kratkočasovna Fourierjeva transformacija. Ta določa frekvenčno in fazno vsebino signala v določenem časovnem odseku. Najprej moramo razdeliti signal na več manjših oken enake dolžine s pomočjo okenskih funkcij in nato na vsakem izračunamo hitro Fourierjevo transformiranko. Rezultat je matrika, ki predstavlja podatke o magnitudi in frekvenci za vsako okno v našem signalu. Problem kratkočasovne Fuorierjeve transformacije je omejitev ločljivosti rezultata. Odločiti se moramo med boljšo ločljivostjo v frekvenčnem ali časovnem prostoru. Široko okno ima boljšo frekvenčno natančnost, a slabšo časovno, ozko okno pa ravno obratno. Tako dobljena matrika nam tvori spektrogram. Spektrogram je najbolj osnovna glasbena vizualizacija frekvenčnega spektra glasbe.



Slika 2.1: Amplitudna ovojnica [7]. Napad (attack) je čas, ki je potreben, da začetni signal zraste od nič do najvišje točke. Razpad (decay) je čas, ki je potreben, da signal pade od najvišje točke do točke ohranjanja (sustain). Sprostitev (release) je čas, ki je potreben, da signal pade s točke ohranjanja na nič.

Ko imamo spektrogram skladbe, lahko z različnimi postopki ločimo iz-

vorne signale iz skupka signalov v skladbi. Eden od pristopov je, da za vsako okno izračunamo povprečno amplitudo sinusoid in jo primerjamo s povprečno amplitudo izvornega signala. Če se ujemata (z neko dovoljeno napako), lahko sklepamo, da je izvorni signal prisoten v tem časovnem oknu. Slabost tega algoritma je, da moramo poznati podatke o izvornem signalu, česar pa velikokrat ne vemo. Lahko pa uporabimo tudi postopke, ki delujejo na podlagi klasifikacije. Signal najprej razdelimo na odseke ter iz vsakega razberemo nekaj osnovnih značilnosti. Značilnost izberemo na podlagi signala, katerega želimo izluščiti (melodija, bobni, šum). S pomočjo značilnosti odseka nato z izbranim klasifikacijskim algoritmom določimo izvor posameznega zvoka. Za ločevanje lahko uporabimo tudi amplitudno ovojnico, s pomočjo katere lahko prepoznamo določene inštrumente, kot so naprimer bobni [27, 17, 30, 21, 22].

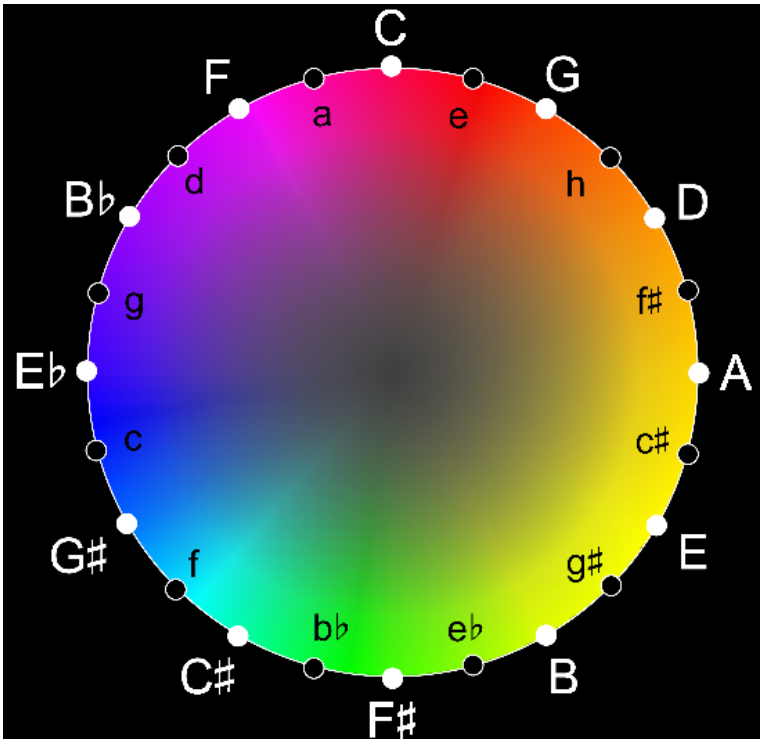
Rezultat transkripcije glasbe je večinokrat MIDI format glasbe. To je standardni elektronski protokol, s katerim lahko komunicirajo različne elektronske glasbene naprave. MIDI format se lahko pridobi s transkripcijo glasbe ali pa se ga napiše na roke. MIDI vsebuje veliko parametrov, med njimi pritisk note, spust note, katera tipka je v uporabi, kako močno pritisnemo tipko, itd [2]. Včasih lahko dobimo že vnaprej pripravljen MIDI zapis skladbe. Skladbo, katero sem si izbral, nima MIDI formata.

Avtomatska transkripcija glasbe je zelo kompleksno in obsežno poglavje računalništva. Primer je diplomska naloga Mihe Pešiča, v kateri je ločeval tolkala od drugih signalov [21]. Skladba, ki sem jo izbral za upodobitev v moji diplomski nalogi, pa vsebuje poleg tolkal še sintetizatorje ter različne šume. Obseg transkripcije bi bil poleg izdelave digitalne animacije za to diplomsko nalogo preobsežen.

2.2 Vizualni model

Ko smo pridobili primerne informacije o skladbi, lahko začnemo z izgradnjo vizualnega modela. Pri izgradnji modela je treba najti dobro preslikavo zvočnega prostora v vidni prostor. Pri vizualizaciji se lahko odločimo za vizu-

alizacijo različnih lastnosti skladbe. Lahko vizualiziramo melodijo, ritem in harmonijo. Vizualizacije lahko uporabljajo barvni prostor ali različno število prostorskih dimenzij.



Slika 2.2: Kvintni krog s preslikavo v barvni prostor [1].

Za vizualizacije harmonije je pogosto, da uporabljajo barvanje tonov po principu kvintnega kroga. Kvintni krog predstavlja ton kot krajevni vektor pozicije v kvintnem krogu. Dve vrsti takih vizualizacij sta Mardirossianova vizualizacija [19] in Sappova vizualizacija[23]. Slabost kvintnega kroga je v tem, da ne zajema povezave med toni, kot so terce. V tem primeru raje uporabimo teroidni model. Naslednji primer je Toivianinenova vizualizacija [25], katera za prikaz harmonij v skladbi uporablja razmerja medtonalitet. Vizualizacije velikokrat uporabljajo RGB barvni prostor. Problem teh prostorov je, da sprememba posamezne vrednosti v tem prostoru ne pomeni iste spremembe v človeški percepciji. CIELab je eden od primernejših barvnih

prostorov za glasbeno vizualizacijo. CIELab bazira na podlagi človeških meritev percepcije barve [15]. Eden od pristopov glasbene vizualizacije je preslikava pozicije tona v kvintnem krogu v izbran barvni prostor. Med preslikavo iz glasbenega prostora v vizualni prostor se moramo vprašati o smiselnosti vizualizacije. Upoštevati moramo, da morajo podobne barve vizualizirati enako podobne glasbene lastnosti. Barvna reprezentacija najmanjših razlik med lastnostmi glasbe se mora dovolj razlikovati, da je vidna in razumljiva. [18].

Problem glasbenih vizualizacij s pomočjo vizualnega modela je ta, da gre pravzaprav za izdelavo vmesnika, ki prevaja zvočne podatke v slikovne podatke, ta vmesnik pa je za vse skladbe enak, zaradi česar so tudi vizualizacije vseh skladb med seboj zelo podobne. Gledalec-poslušalec tako ne dobi drugačne podobe za povsem različne skladbe. Samo podobo zaradi tega doživlja kot redundantno (ne prinaša novih informacij, ki niso že vsebovane v sami skladbi) in generično (vse skladbe imajo podobno podobo). Moj namen pa je bil, da vzamem skladbo kot celoto, ne pa zgolj kot nek signal. Rezultat ni kvantitativna uprizoritev ampak kvalitativna interpretacija, kognitivna izkušnja skladbe.

2.3 VJ-janje

Ker sem naredil vizualno reprezentacijo skladbe, ki komplementira izbrano elektronsko skladbo in bi se lahko uporabljala tudi na predstavah v živo, lahko še omenim podobno artistično obliko, imenovano VJ-janje. Vj-janje je akronim za video jockey in prihaja iz fraze DJ – disc jockey. Tako kot disc jockey miksa glasbene plošče v živo pred občinstvom, tako tudi video jockey "miksa" videe v živo. VJ-janje poteka večinoma v diskotekah ter na koncertih, kjer tako imenovani VJ-i miksajo in spreminjajo posnetke na podlagi glasbe, ki se trenutno vrti. S tem povečajo učinek glasbe na občinstvo ter ustvarjajo vizualno-zvočni ambient. VJ-janje ima dolgo zgodovino, populariziral pa ga je televizijski kanal MTV v 80. letih. Takrat je Merrill Aldighieri

v klubu v New Yorku eksperimentirala z videoposnetki, ki so dobro komplementirali glasbo. Njeno metodo VJ-janja je sestavljalo improviziranje s posnetki ter kratkimi ponavljajočimi se filmski odseki. Med programi, ki se danes uporabljajo za VJ-janje, so VDMX, ProVideoPlayer 2, Resolume in podobni. Na podoben način sem tudi jaz naredil svojo vizualno reprezentacijo skladbe, vendar v mojem primeru posnetek ni narejen v živo. Če bi recimo hotel DJ, od katerega sem uporabil glasbo, narediti tudi VJ-šov, bi moral moje posnetke izrezati in jih v realnem času lepiti oz. jih spreminjati, kar pa ne bi bilo težko, saj ima celotna skladba ves čas isti ritem, in bi bili prehodi med samimi posnetki videti popolnoma sinhroni [32, 33].

Slika 2.3: VJ-janje [11].

Poglavje 3

Pregled orodij za digitalno animacijo

Dandanes imamo veliko število različnih 3D-programov in paketov, zato je zelo pomembno, za katerega se bomo odločili. Razlikujejo se v ceni, kakovosti in dostopnosti. Na eni strani imamo odprtokodne programe, katerih kodo lahko vidimo brezplačno in tudi mi prispevamo k razvoju. Na drugi strani pa imamo licenčne programe, ki se uporabljajo za zelo specifične naloge ter lahko stanejo več tisoč evrov. Programi se predvsem razlikujejo v specializaciji. Nekateri so specializirani za filmske efekte, drugi za industrijo iger, tretji za animirane filme, četrti pa za realno simulacijo naravnih pojavov.

Za animacijo, katero sem imel v mislih, sem iskal program po naslednjih kriterijih:

- **Dostopnost ter kompleksnost orodja.** Pri programih lahko izbiramo med takimi z veliko funkcionalnostjo, a hkrati veliko kompleksnostjo, in takimi z manjšo kompleksnostjo in hkrati manjšo funkcionalnostjo. Sam sem iskal program, ki ima dovolj veliko funkcionalnost, a hkrati ni preveč kompleksen.
- **Uporabniški vmesnik.** Na začetku sem hotel narediti animacijo v programu Maya, ampak me je odvrnil zastarel in nepregleden upo-

rabniški vmesnik programa. Če ima program pregleden uporabniški vmesnik, lahko veliko hitreje naredimo animacijo, kot pa če je nepregleden.

- **Kompatibilnost.** Različni programi delujejo na različnih operacijskih sistemih.
- **Specializacija programa.** Večina programov na trgu ponuja podobne funkcionalnosti. Programi se razlikujejo v nekaj manjših, a ključnih stvareh, za katere so specializirani.

3.1 Blender

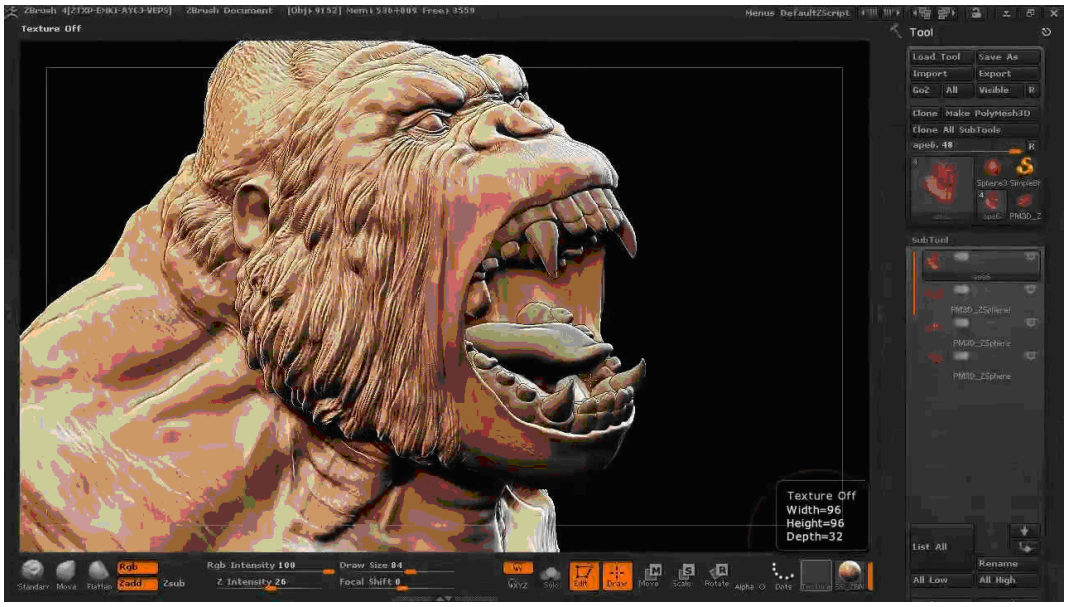
Blender je odprtokodni 3D-program za animacijo, modeliranje, vizualne učinke, umetnost, interaktivne aplikacije, video igre in 3D-tisk. Njegova največja prednost je njegova cena, saj je zastonj. Njegove zmožnosti so zelo dobre, vendar je lahko zahteven za učenje. Posledica tega, da je brezplačen, je tudi ogromno število virov na spletu.

3.2 Zbrush

Zbrush je 3D-program za modeliranje ter digitalno kiparjenje. Je odličen za ustvarjanje modelov visoke resolucije za igre, animacije in filme. Najbolj znan je po tem, da lahko rišemo po 3D-modelu in ga "kiparimo" kar na roke, kot bi oblikovali glino. To nam predvsem pomaga pri organskih površinah.

3.3 SolidWorks

SolidWorks je program za 3D-modeliranje oz. CAD (computer aided design). CAD-programi imajo poudarek na natančnih ilustracijah tehnične narave, na primer izdelkov, ki gredo pozneje v proizvodnjo. Uporabljajo ga arhitekti, inženirji, oblikovalci.



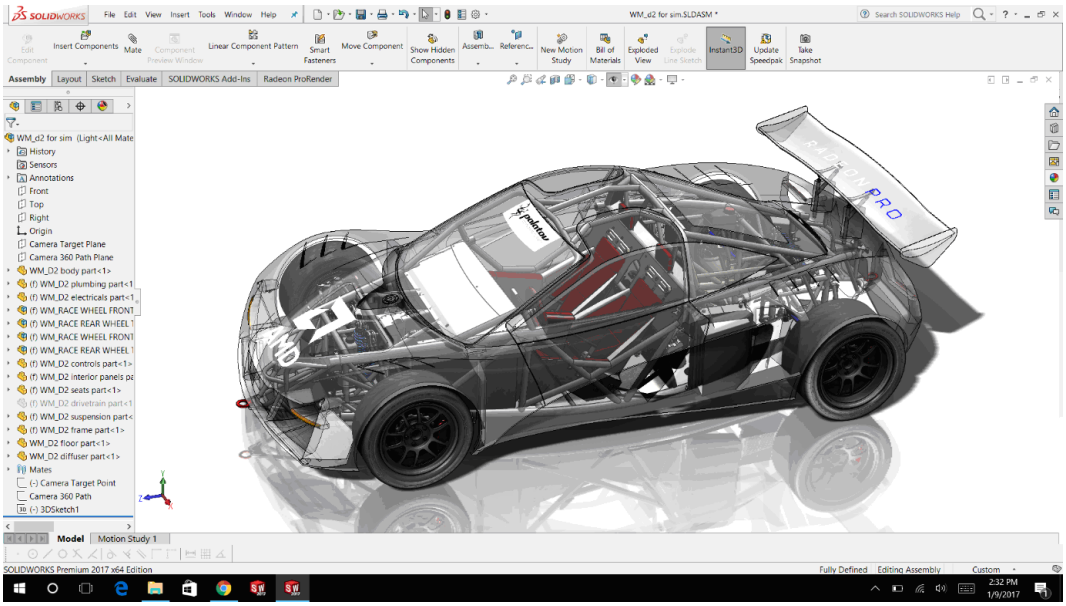
Slika 3.1: Uporabniški vmesnik programa Zbrush [12].

3.4 Maya

Maya je eden izmed najbolj popularnih 3D-programov. Podpira modeliranje, animacijo in vizualne učinke. Uporablja se predvsem v filmih, animiranih filmih in za vizualne učinke. Program Maya je združen s paketom Mental Ray, katerega naloga je foto-realistično upodabljanje. Maya je lahko prilagojena z uporabo vtičnikov tujih proizvajalcev, ki so napisani v Mayini različici jezika Python, jeziku MEL (Maya embedded language).

3.5 3ds Max

3ds Max je 3D-program, ki je kot Blender in Maya zmožen vsega. Uporablja se predvsem v industriji računalniških iger in arhitekturi. Pri slednji se uporablja predvsem zaradi hitrega in dobrega modeliranja. Njegova orodja za animacijo niso tako izpopolnjena kot v programu Maya, pač pa se izkaže pri modeliranju in teksturiranju. Tako kot Maya ima tudi 3ds Max razvit svoj



Slika 3.2: Uporabniški vmesnik programa SolidWorks [10].

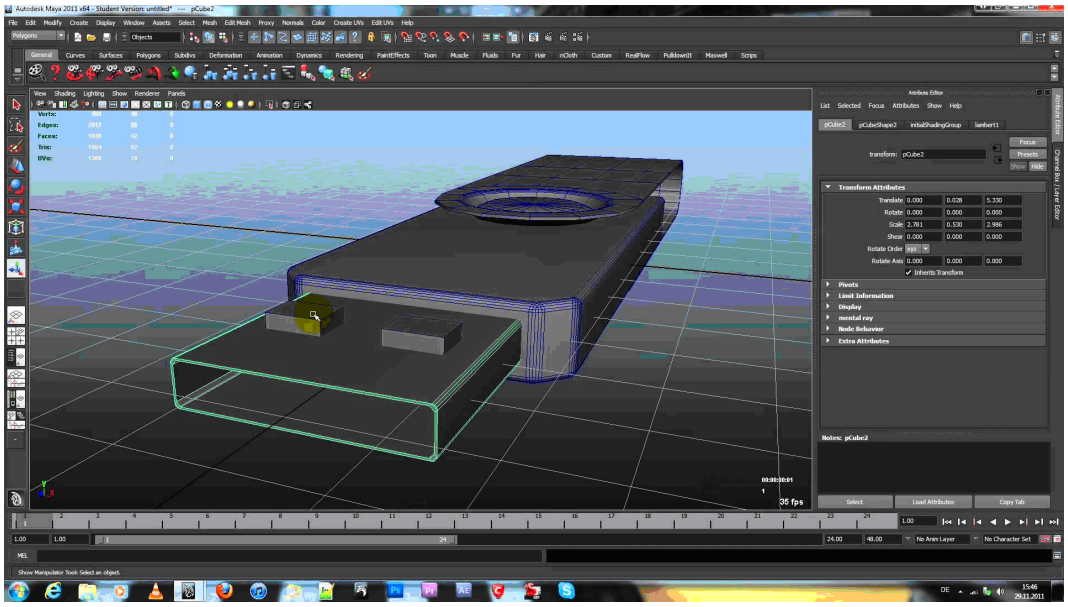
jezik, imenovan MAXScript. Velika slabost programa 3ds Max je v tem, da je podprt samo na Windows platformi, za razliko od Maya, ki je podprta na Windows, Linux ter OSX [28].

3.6 Houdini

Houdini je eden od glavnih 3D-paketov za proceduralno razvijanje okolja. Dober je pri fizikalnih simulacijah delcev in tekočin ter simulacijah gibanja množic. Zelo je popularen pri produkcijskih hišah za vizualne učinke, kjer je prototipiranje ključnega pomena. Deluje na podlagi vozlišč, ki se z lahkoto kopirajo in spreminjajo.

3.7 Cinema4D

Cinema4D se uporablja predvsem v zabavni industriji, animacijah, reklamah, animiranih in odzivnih znakih in logotipih, gibljivi grafiki, v televizijskih

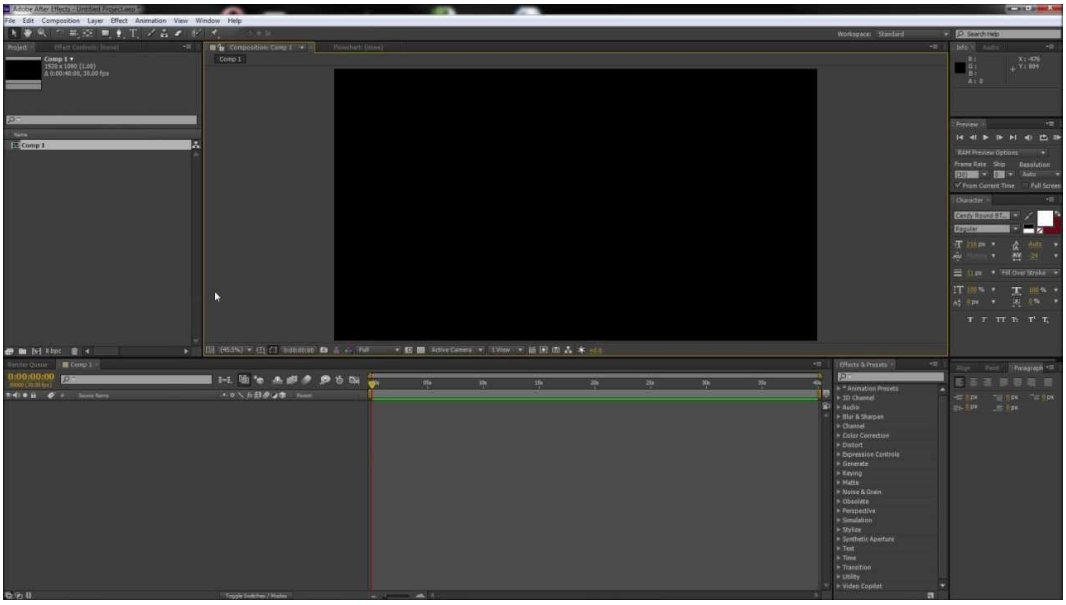


Slika 3.3: Uporabniški vmesnik programa Maya [8].

špicah in pri umetniških projektih. Ima zelo čist in razumljiv uporabniški vmesnik ter je zato zelo privlačen za začetnike. Lažje se je naučimo kot na primer programov Maya ali 3ds Max, vendar ima slabša orodja za modeliranje in animacijo. Njena moč leži predvsem v gibljivi grafiki, proceduralnem modeliranju ter animaciji. Cinema4D uporablja dva jezika, integrirani Cinema4D jezik, imenovan COFFEE, in Python. Kratica COFFEE ne pomeni ničesar, temveč je samo duhovita referenca na jezik Java. Z obema jezikoma lahko dosežemo podobne rezultate. Cinema4D uporablja tudi močan sistem vozlišč Xpresso, kjer so objekti v sceni predstavljeni kot vozlišča, katerih vhode in izhode lahko povezujemo. Poleg tega lahko tem vozliščem napišemo tudi svojo kodo. Cinema4D ima zelo dober vtičnik CINEWARE, kateri omogoča neprekinjeno integracijo med Cinema4D in Adobe After Effects CC [29].

3.8 Adobe After Effects

Adobe After Effects je program specializiran za 2D-animacijo, post-procesiranje, vizualne učinke, gibljivo grafiko ter video montažo. Uporablja se v filmski in televizijski industriji ter animaciji [20, 3, 14].



Slika 3.4: Uporabniški vmesnik programa After Effects. V spodnjem delu uporabniškega vmesnika imamo časovnico, kjer lahko razporejamo upodobljene posnetke iz Cinema4D v steze. Vsaki stezi lahko dodamo poljubno število učinkov, katerim parametre spreminjamo na desni strani vmesnika. Parametre lahko tudi animiramo s pomočjo krivulj v časovnici. Vsaka steza ima plasti, ki omogočajo lažje urejanje končnega izdelka.

3.9 Zakaj sem se odločil za Cinema4D

Program Cinema4D je prijazen do začetnikov. Ima lep in čist uporabniški vmesnik. Za razliko od programov Maye ali Zbrush, kjer sem kakšno funkcijo iskal tudi po 10 minut, ker je bila skrita na kakšnem majhnem gumbu, Ci-

nema4D poskrbi, da je vsaka funkcija razvidna, velikost in pozicija gumbov sta konsistentni in font je dovolj velik. Če primerjamo to z industrijskim standardom – Maya, ki ima razmetan uporabniški vmesnik, in Zbrushem, ki ima najbolj unikaten in zato frustrirajoče drugačen uporabniški vmesnik, vidimo, zakaj je program Cinema4D tako privlačen za začetnike. Res da Cinema4D ni tako močna v animacijskem ali celo v modelirnem oddelku, je pa zato zelo dobra v proceduralnih efektih, ki lahko spreminjajo skoraj vse v realnem času – od oblik 3D-objektov do njihovega premikanja, parametrov materialov in tako dalje. In ravno take proceduralne efekte sem imel v mislih za svojo diplomsko nalogo. Nisem potreboval vpenjanja okostja v animirane like ali dobrega orodja za izdelovanje tekstur ali celo dobrih modelirnih orodij. Uporabil sem osnovne 3D-oblike in jih spreminjal, animiral ter spreminjal na podlagi proceduralnih efektov. Cinema4D ima odlično dokumentacijo kar v samem programu, saj ima pri vsakem okencu in vsakem orodju gumb za pomoč. Ta nam odpre okno z detajlnim opisom orodja, primeri uporabe in slikami ter povezavami do uporabnih virov na spletu. Dodal bom še to, da sem za program Maya našel samo en vtičnik, ki omogoča spreminjanje parametrov na podlagi frekvence, in nič globljega. Cinema4D pa ima to orodje že v samem paketu in ima več možnosti nastavitev, od omejevanja frekvence ter kompresije. Tako sem že na začetku opazil veliko razliko med tema dvema programoma in njunimi ciljnimi uporabniki. Poudarek programa Cinema4D je animacija, ki se ne izdeluje ročno, pač pa je večinoma narejena proceduralno oz. dinamično. Ker sem imel v mislih tovrstno animacijo, ter zaradi boljše podpore in integracije vizualizacije glasbe, sem se odločil za Cinema4D. Za post-procesiranje, montažo ter vizualne efekte sem izbral program Adobe After Effects, ki je vodilni v tej panogi.

Poglavje 4

Uporaba orodja Cinema4D

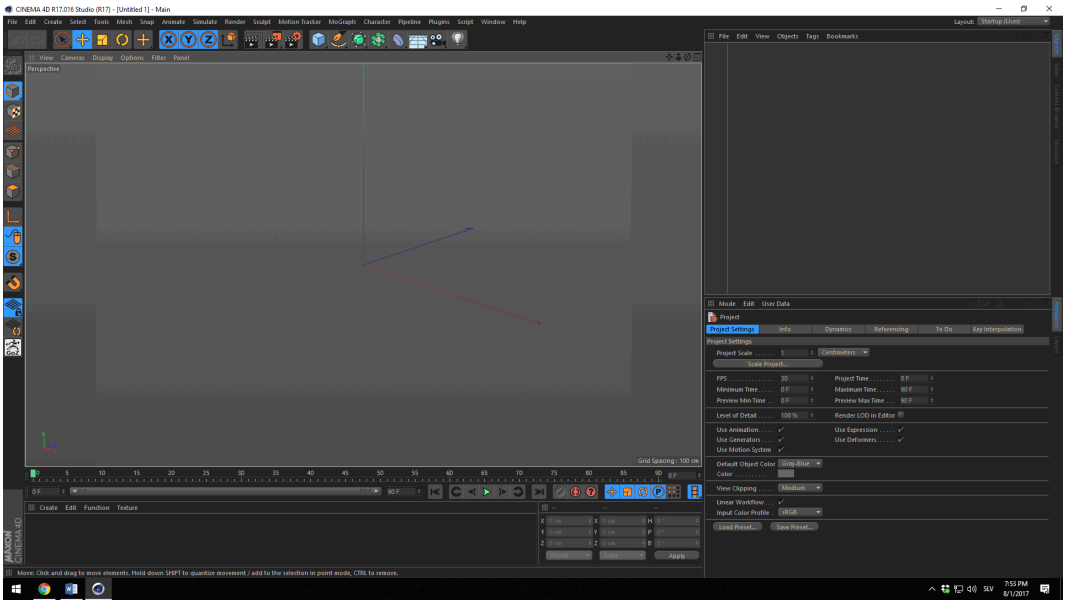
Zakaj se Cinema4D imenuje Cinema4D? Kot vemo, imamo tri različne dimenzije: višino, širino, dolžino. Nekatere teorije pravijo, da je četrta dimenzija čas. Ker je bila Cinema4D ustvarjena za animacije, ni presenetljivo, da so jo avtorji poimenovali po četrti dimenziji, času. Cinema4D je razvilo podjetje pod imenom MAXCON Computer Gmbh v Nemčiji. Zgodba programa sega v leto 1990, v katerem sta se Christian in Philip Losh udeležila Kickstart tekmovanja in ga osvojila. Prva verzija Cinema4D sega v leto 1993 in je bila razvita za računalnike Amiga. Po propadu Commodorja so se poznejše verzije Cinema4D osredotočale na Windows in iOS operacijske sisteme. Cinema4D so uporabili v znanih filmih, kot so Človek-pajek 3, Zgodbe iz Narnije, Polarni vlak, Kralj Artur, Tron: Zapuščina in podobni [4, 5].

4.1 Pregled vmesnikov in orodij v Cinema4D

Če odpremo Cinema4D, vidimo na ekranu približno tako sliko:

Na sredini imamo projektni pogled (project view). V tem okencu vidimo našo sceno, 3D-objekte v njej, kamere, luči, krivulje, učinke in delce. S pomočjo okenca pozicioniramo objekte v sceni, rotiramo, povečujemo, animiramo in teksturiramo. To je naš predogled končnega izdelka.

Na desni imamo upravljalnik objektov (object manager), kjer so prikazani



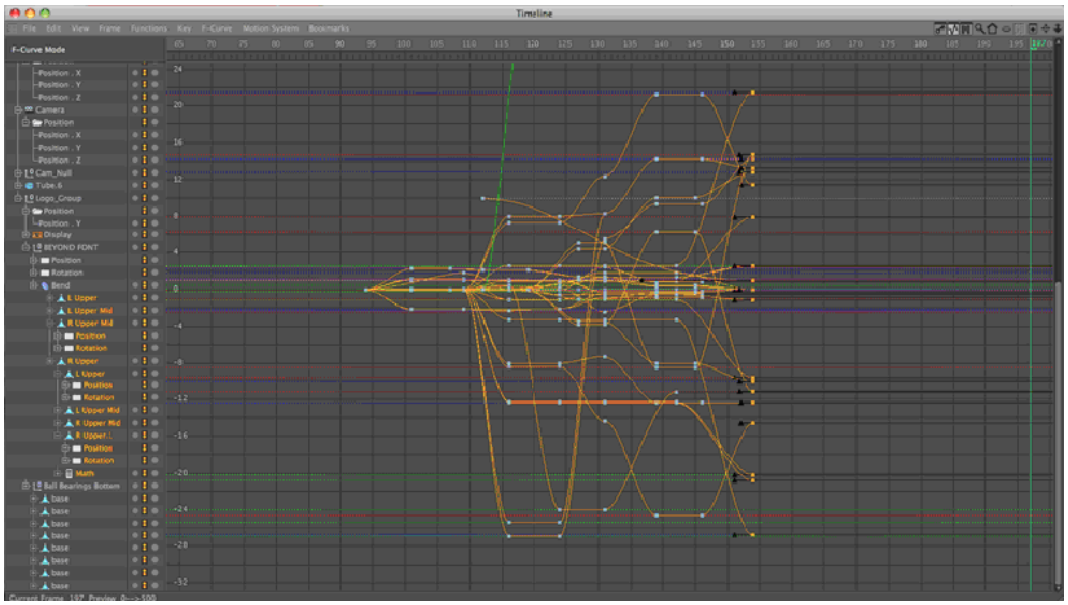
Slika 4.1: Uporabniški vmesnik programa Cinema4D.

vsi objekti v trenutni sceni. Razdeljeni so v hierarhijo – oče–sin. V tem okencu vidimo materiale objektov, skripte ter efekte, ki jih imajo na sebi. Desno spodaj imamo okno atributov. Tukaj lahko vidimo atribute izbranega objekta. Vsak atribut lahko spreminjamo ter animiramo po želji.

Zgoraj imamo vrstico z meniji, malo pod njo pa bližnjice najbolj pogostih funkcij. Od leve proti desni so nanizane običajne funkcije, ki so v katerem-koli 3D-programu: premikanje, rotiranje, povečevanje, zaklepanje različnih koordinat, upodabljanje, nastavitve upodabljanja, primitivni objekti, krivulje, mograph proceduralni efekti, zaradi katerih je Cinema4D tako znana, kamere in luči. Na levi imamo orodja za izbiro, kot so točkovna selekcija, robna selekcija in ploskovna selekcija, in orodje, ki pomaga izbirati in razporejati objekte (snap tool). Spodaj imamo urejevalec materialov, kjer so prikazani vsi materiali v sceni, malo nad njim pa je časovni trak v okvirjih na sekundo. Kot vidimo, je UI zelo čist in jasen. Okenca lahko poljubno premikamo in raztegujemo.

Omembe vredna je še časovnica f-krivulj [6], katere namen je animacija

katerihkoli parametrov s krivuljami. Določimo lahko tip krivulje in točke, tem točkam pa čas na časovnici. 3D-objekti se spreminjajo na podlagi krivulj, kot so prikazane naprimer na sliki spodaj.

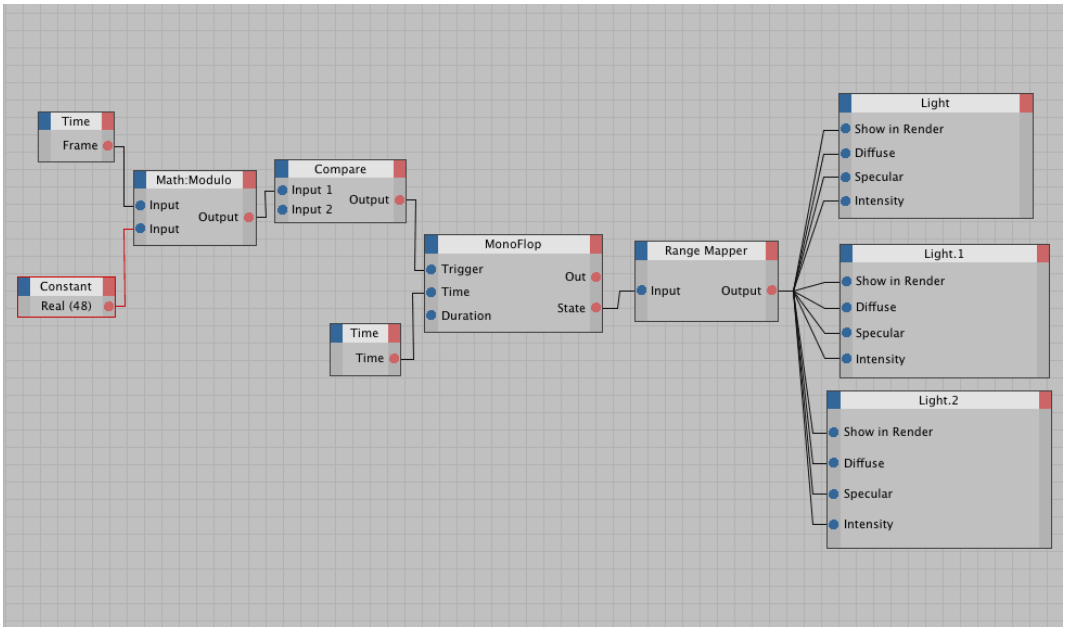


Slika 4.2: Slika prikazuje časovnico f-krivulj. Na levi je seznam objektov, katere animiramo s pomočjo točk in krivulj na desni strani okna. [6].

Pomemben del Cinema4D je tudi urejevalnik Xpresso [9], s katerim se da voditi parametre efektov na podlagi posebnih skript ali pa drugih objektov v Cinema4D. Urejevalnik Xpresso deluje na principu vozlišč. Ta predstavljajo bodisi konstanto, skripte z vhodi in izhodi, ter dejanske 3D-objekte v sceni. Vhodi so obarvani modro, medtem ko so izhodi obarvani rdeče, povezujemo jih lahko kar s pomočjo UI-ja. S pomočjo tega orodja lahko vizualno sprogramiramo zanimive učinke, ki bi jih lahko v drugih programih sprogramirali samo s kodo.

Cinema4D ima tudi zelo dober urejevalnik materialov. Materialu lahko izberemo barvo, difuzijo, sevanje, prozornost, odboj, teksturo, reliefnost, mapiranje prestavitve in podobno. Vse naštete parametre lahko animiramo.

Kot sem že omenil, se moč Cinema4D skriva v učinkih Mograph, ki so

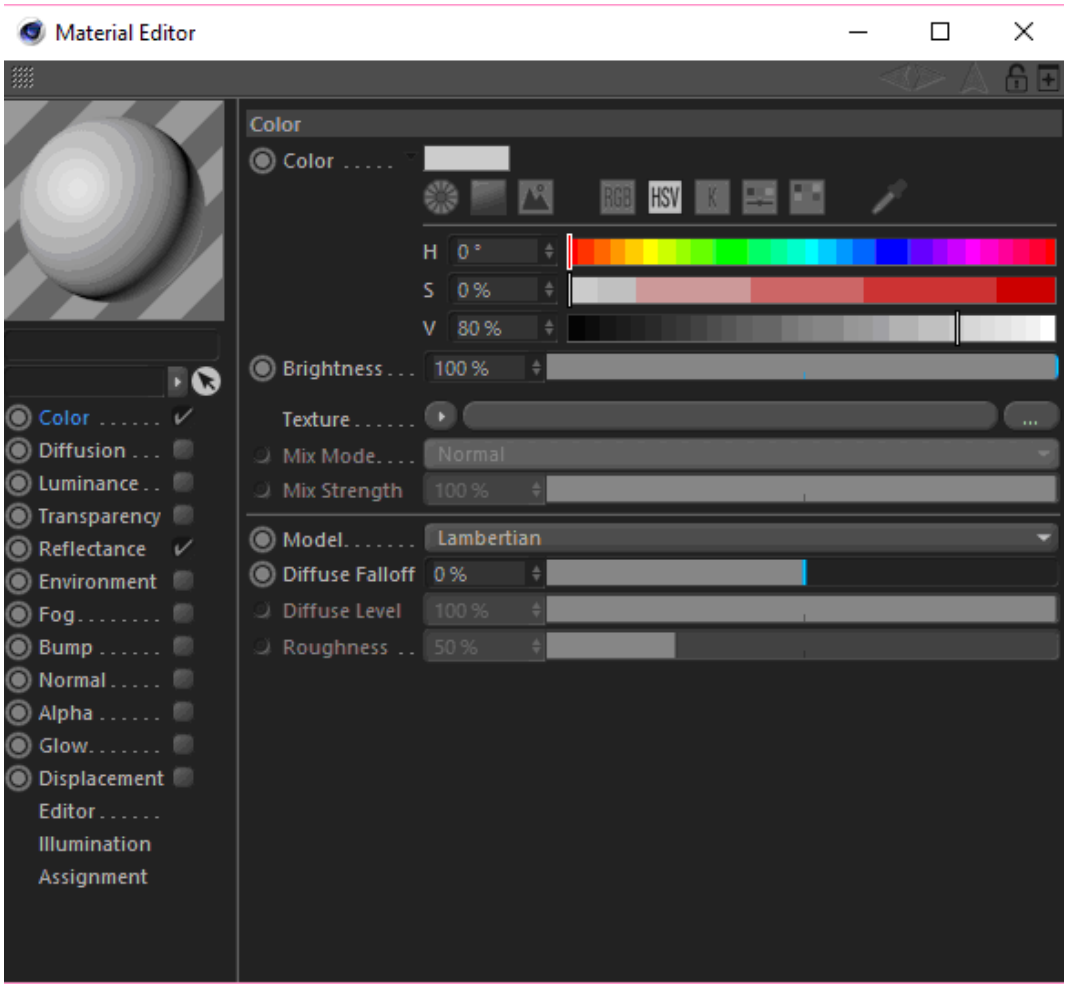


Slika 4.3: Uporabniški vmesnik urejevalnika Xpresso [9]. Kvadrati predstavlja objekte v Cinema4D. Povezave med njimi predstavljajo odvisnost med parametri različnih objektov.

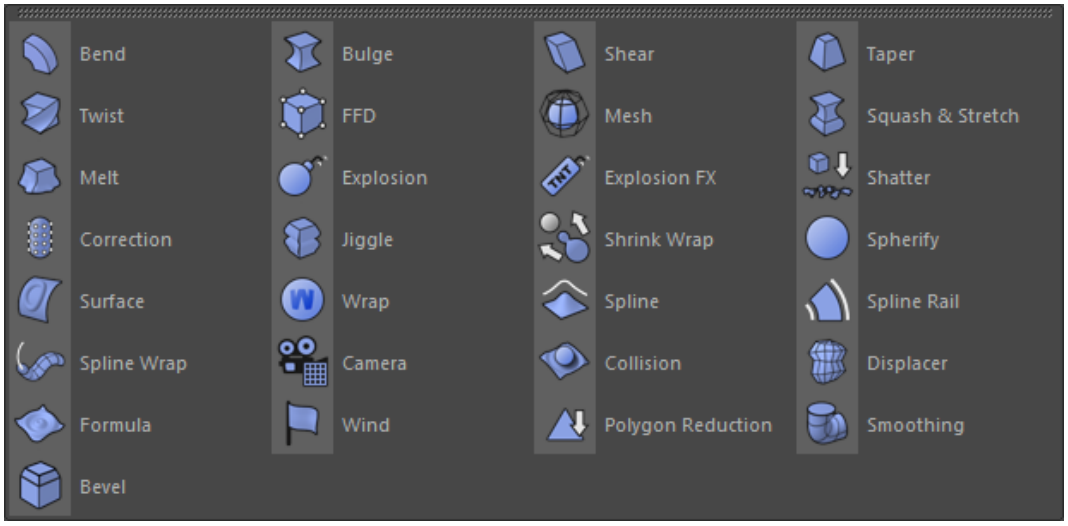
dejanski 3D-objekti, katerih oblika je lahko poljubna (tudi neskončna). Sami po sebi so nevidni, a ko jim določimo objekt učinkovanja, dobimo zanimive rezultate. Eden najboljših je učinkovalec (effector) kloniranja, ki klonira katerekoli objekte na poljubno pozicijo, objekt ali krivuljo. Ker lahko učinkovalec učinkuje tudi na drug učinkovalec, lahko na ta način zgradimo kompleksne objekte in animacije. Možnosti učinkovanja so številne in raznolike, od objektov, ki privlačijo dinamične objekte, do razpada ter ukrivljenja 3D-objekta.

Omeniti velja tudi zvočni učinkovalec (sound effector), ki je specializiran za spreminjanje objektov in animacije s pomočjo analize zvočnih datotek. Vanj vstavimo glasbeni format wav in tako dobimo grafično razviden frekvenčni graf. Frekvenčni graf lahko omejimo s frekvenčnim filtrom in ga tudi kompresiramo. Kompresija je postopek večanja amplitude manjših frekvenc, a hkrati manjšanje amplitude večjih frekvenc, zato da dobimo bolj

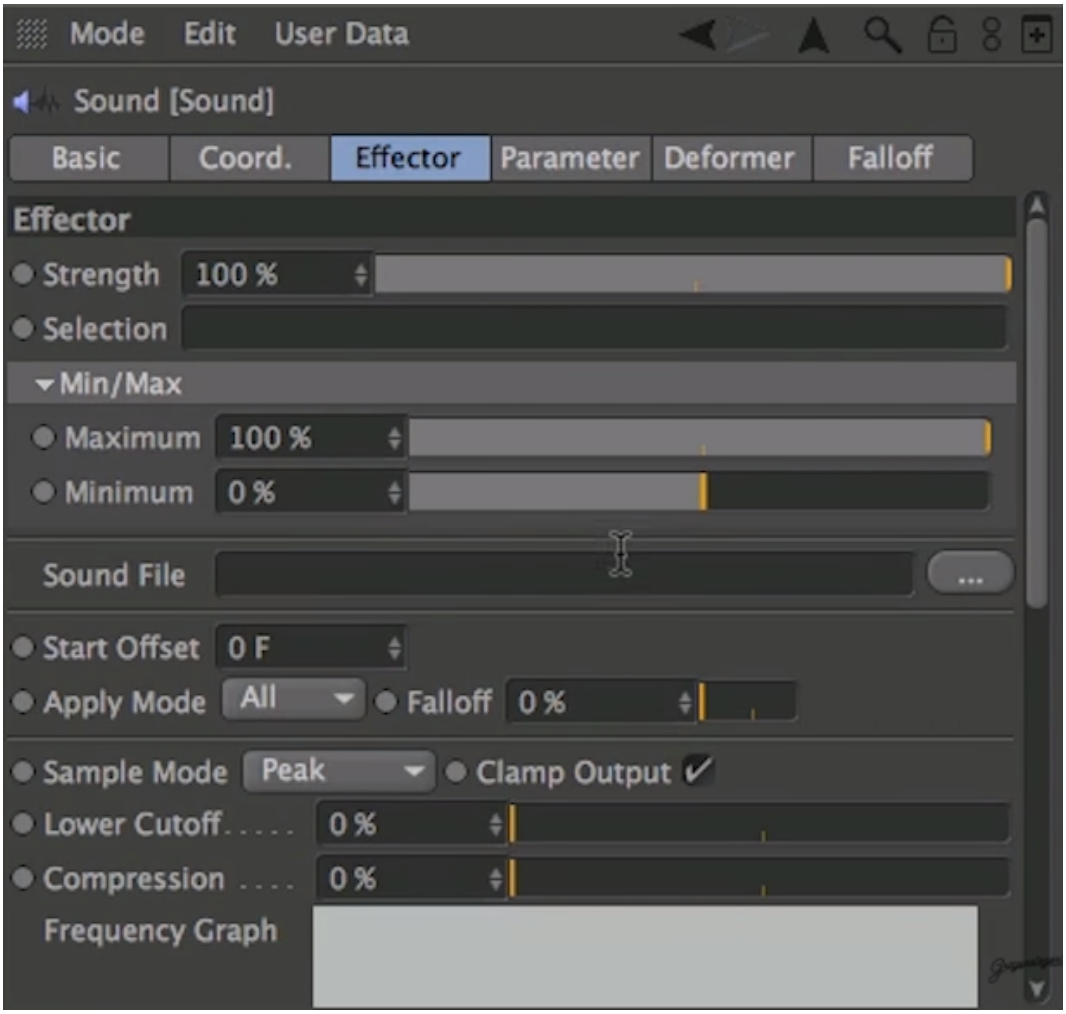
enakomerno glasnost vseh zvočnih signalov. Z različnimi vrednostmi v frekvenčnem grafu lahko nato spreminjamo objekte in vrednosti v Cinema4D. Naštete učinkovalce sem uporabil v svoji digitalni animaciji.



Slika 4.4: Uporabniški vmesnik urajevalca materijala. Na levi vklopimo različne lastnosti materijala. Na desni pa spreminjamo parametre vsake označene lastnosti.



Slika 4.5: Vsi razpoložljivi učinki Mograph.



Slika 4.6: Zvočni učinkovalec. Med parametri vidimo moč, zvočno datoteko, frekvenčni graf, kompresijo in filtriranje nizkih frekvenc.

Poglavje 5

Postopek izdelave digitalne animacije

Digitalno animacijo, ki sem jo izdelal v okviru diplomske naloge, sem razdelil na štiri dele, ki sledijo delitvi v glasbeni podlagi izbrane skladbe. Vsakega od delov skladbe sem vizualiziral drugače. Prvi del, ki naj bi prikazal svet, kot ga vidimo mi, torej svet, ki ga lahko izmerimo v metrih, drugi del, v katerem kamera potuje na celično raven, torej svet lahko merimo v mikrometrih (10^{-6}), in nato tretji del, kjer potujemo v svet atomov in nanometrov (10^{-9}), ter celo globlje od atoma, kjer naj bi po teoriji superstrun prebivale strune velikosti plank mere (10^{-35}). V končnem, četrtem delu videa pa gremo v svet velikega, torej svet vesolja, kjer si oblike lahko predstavljamo v megametrih (10^{6}) ali več. V vsakem delu diplomske naloge sem hotel pokazati razgibanost in fascinantno okolje narave ter znanosti, ki ga ne vidimo vsak dan. Zanimivo je tudi povedati, da je avtor skladbe Singularity (z albuma Powers of Ten) Stephan Bodzin inspiracijo dobil iz knjige, ki govori o velikosti in razsežnosti vesolja (Powers of ten avtorja Philipa Morrisona). Knjiga bralca vodi skozi različne velikosti dimenzij, od bilijon svetlobnih let do sveta atoma. Moja ideja ne izvira iz inspiracije njegovega albuma, saj te informacije nisem imel, ko sem začel delati diplomsko nalogo. Inspiracijo sem namreč dobil od poslušanja Bodzinove glasbe. Šele pozneje sem se zavedel, da se je ob njegovi

glasbi rodila ta ideja, kot pri njemu ob branju knjige.

Izdelave 3D-animacije se lotimo po naslednjem postopku:

- **Raziskava področja ter referenčnega materiala.** Pregledamo področje glasbene vizualizacije ter sorodne animacije, iz katerih lahko črpamo inspiracijo.

- **Izvirna ideja.** S pomočjo metode *brainstorming* se prepustimo toku asociacij in tako pridobimo množico idej, iz katere izberemo najboljšo.

- **Izbira potrebnih orodjih ter pristopov.** Pregledamo orodja na trgu in izberemo najprimernejša za naše potrebe.

- **Načrtovanje ideje z upoštevanjem tehničnih omejitev programa.** Planiramo postopek izdelave animacije s pomočjo orodja, ki smo ga izbrali, in njegovih omejitev.

- **Izdelava animacije v programu.** Izdelamo animacijo v izbranem orodju.

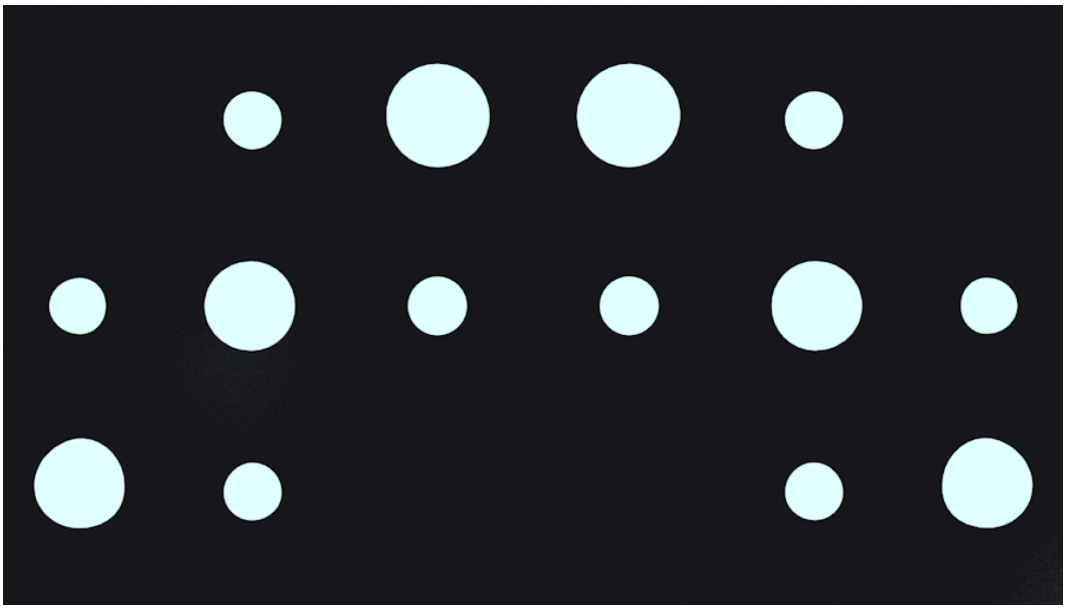
- **Upodabljanje animacije.** Računalnik preračuna končno upodobitev animacije. To je zelo dolgotrajen postopek, ki lahko traja tudi več dni.

- **Iterativno ponavljanje izdelave in upodabljanja animacije, dokler ne pridemo do končnega izdelka.** Velikokrat nismo zadovoljni s končnim izdelkom, katerega vidimo šele po upodabljanju animacije. Takrat se moramo vrniti k izdelavi animacije ali pa celo malo spremenimo načrt animacije, da pridemo do želenega končnega izdelka.

5.1 Izdelava digitalne animacije v prvem delu videa: kloniranje objektov, ustvarjanje razlik z zamiki in deformacijami

V tem delu sem skušal prikazati svet, ki ga lahko vidimo ljudje. Animacija predstavlja planet, po katerem potujemo. Najprej sem ustvaril osnovno primitivno obliko, kroglo. Eno večjo, temnejšo, ki naj bi predstavljala planet, in drugo manjšo, svetlejšo, ki predstavlja majhne krogle na planetu. Nato sem ustvaril učinkovalec kloniranja (cloner effector). Nastavil sem mu, naj klonira objekte na vsako točko velike krogle. Za sina v hierarhiji sem mu nastavil kroglo ter tri ničelne objekte. Tri ničelne objekte sem dodal zato, da bo za vsako kroglo, ki jo bo kloniral, kloniral še tri prazne objekte. Tako se izognemo temu, da bi kloniral na vsako točko krogle, s tem dobimo bolj redko poseljenost krogle. Izjema temu je začetna scena, kjer sem kloniral na vsako točko posebej. Ker sem želel, da začetne krogle ne povzročijo odseva na planetu, sem jim dodal oznako kompozicije (composition tag), v kateri lahko označiš, da se objekt, ki ima to oznako, ne pojavi v odsevih drugih objektov. Tako sem dobil 2-dimenzionalni videz začetne scene. Ker sem hotel, da krogle lebdijo nad površino, sem pri parametrih kloniranja nastavil zamik po z-osi. Vendar to ni bila najboljša rešitev, ker je kloniranje vse klonirane krogle premaknilo po globalni z-osi, in ni upoštevalo rotacije dejanskih krogel. Jaz pa sem hotel, da bi vsako odmaknilo od centra krogle. Ta problem sem rešil tako, da sem kloniral ničelne objekte, ki so imeli za sina te krogle. Vsaki krogli sem nato nastavil zamik po lokalnem z-ju, ki je upošteval orientacijo očeta. Ker sem hotel, da krogle lebdijo, sem vsaki krogli dodal animacijo lebdenja. Prišlo je do problema, da so vse krogle lebdele istočasno, jaz pa sem hotel naključne začetke in konce animacij. To sem rešil z učinkovalcem naključja, kateri je zamaknil animacijo vseh kloniranih krogel za nekaj okvirjev na sekundo, kar je posledično povzročilo, da so vse krogle lebdele po različnem časovnem intervalu.

Že pri tako majhni sceni sem zaradi velikega števila točk krogle in de-



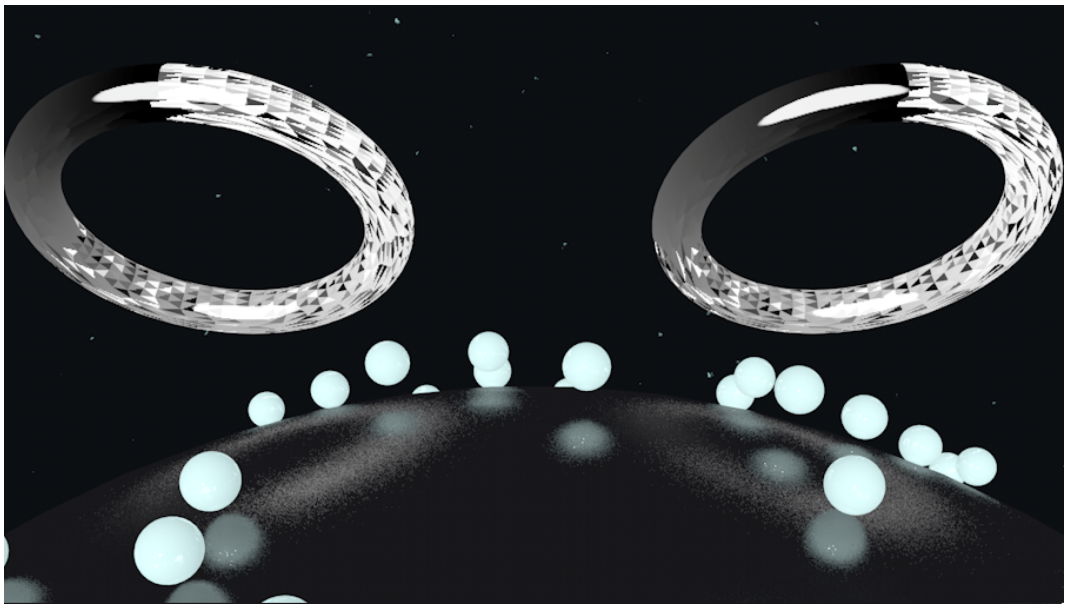
Slika 5.1: Začetni izris krogel, kateri nam daje 2-dimenzionalni videz

janskih krogel, ki jih kloniram na to kroglo, izgubil pri optimizaciji (povečal se je čas upodabljanja). Da bi izboljšal optimizacijo, sem naredil dve veliki krogli, eno, ki ima majhno število točk, in drugo, ki ima veliko število točk. Kroglo z velikim številom točk sem nastavil na malo večji radij, tako da se je v končni upodobitvi videla vedno samo velika krogla. Učinkovalcu kloniranja sem naročil, naj klonira na kroglo z majhnim številom točk, tako sem lahko zreduciral veliko število majhnih krogel. Uporabil pa sem tudi orodje za izbiro točk (point selection tool). S tem orodjem sem označil točke in s tem še dodatno omejil pogoje za nastanek kloniranih krogel. To selekcijo sem nato povezal v objekt kloniranja (cloner object), da je učinkovalec krogle kloniral samo na te točke.

Ker sem hotel, da na začetni sceni vidimo krogle brez praznih prostorov med njimi, sem to rešil tudi s tem orodjem za izbiranje. Začetnim kroglam sem animiral velikost na roko in jim nastavil tako ponavljanje velikostnih sprememb, da so bile animacije v sinhronizaciji z glasbo. To, da so ob nenadnih glasbenih učinkih še dodatno zavalovale, sem naredil z učinkovalcem za-

mude (delay effector), ki sem mu nastavil atribut elastičnosti (spring mode). To je povzročilo samodejno valovanje objektov, kar pripomore k bogatejšemu vizualnemu učinku. V začetni sceni sem uporabil tudi učinkovalec izbokline (bulge effector), ki ga opazimo, ko potuje od leve proti desni in spreminja krogle. Učinkovalec izbokline objekt skrči ali pa ga raztegne, ko je v njegovem radiju. Za deformacijo velike krogle sem uporabil deformator formule (formula deformer). Ta na podlagi matematične funkcije deformira celotno površino objekta.

Kamero sem premikal po krivulji v obliki krogle, katere center sem nastavil na center velike krogle, in radius malo večji od te krogle. Ker sem hotel, da kamera vedno gleda na površino krogle, sem uporabil oznako omejitve (constraint tag). Ta nam omogoča, da omejimo objekt na podlagi parametrov. V



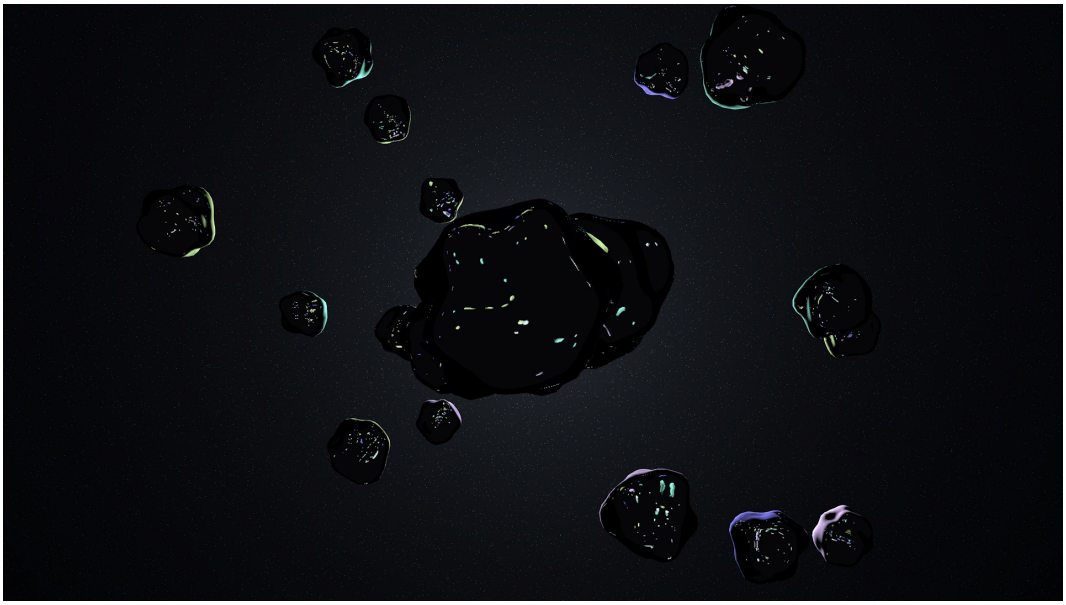
Slika 5.2: Naključno klonirane krogle in obroča, njun videz je spremenjen s pomočjo redukcije poligonov.

mojem primeru sem omejil rotacijo kamere na točko centra krogle, z zamikom 45 stopinj. Rotacija dveh obročev, ki se pojavita na polovici, je animirana na roko. Učinek, ki ga imata na sebi, pa se imenuje redukcija poligonov (po-

lygon reduction). Ta učinek zreducira poligone na objektu. Kako močno jih zreducira, je odvisno od moči parametra, ki sem ga animiral. Končna krogla, ki jo vidimo kot črno zrcalo, je animirana na podlagi učinkovalca naključja. Ta randomizira skoraj katerikoli parameter. V tem primeru sem uporabil učinek za spreminjanje pozicije točk krogle. Materiali, ki sem jih uporabil v tem kadru, so uporabljali kanale, ki skupaj tvorijo material v Cinema4D. Ti kanali so recimo kanal fiksne barve ne glede na luči, kanal prozornosti, odsevni kanal ter številni drugi. Materiale sem animiral na različnih mestih scene.

5.2 Drugi del videa: izvor delcev

V tem delu diplomske naloge sem skušal pokazati celično raven. Prikazuje celice, ki se premikajo vsaka po svoje ter prihajajo proti nam.



Slika 5.3: Krogle različnih barv, katere oddajam s pomočjo oddajnika.

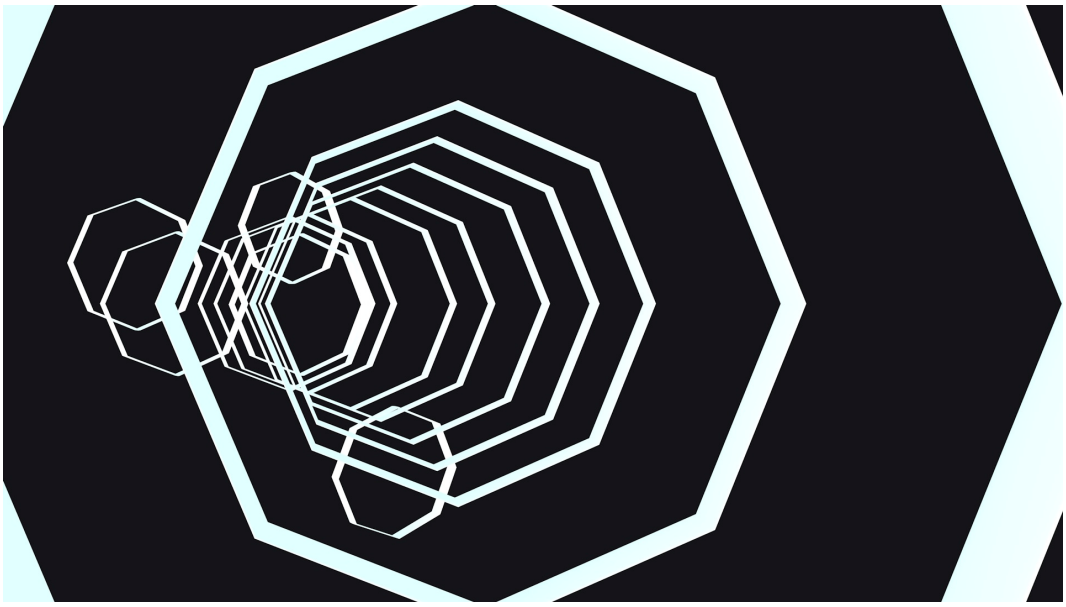
Ozadje sem naredil s preprosto ravnino, ki sem ji dodal teksturo šuma. Tej teksturi sem spremenil barvo šuma, da ni črno-bel in se bolj ujema z

drugimi objekti v moji sceni. Šumu sem še spremenil velikost, da ni deloval preveč redek ali pa preveč gost. Na sredino sem dodal luč, ki odseva svetlobo v vse smeri enakomerno. Za sistem delcev (particle system) sem uporabil standardni izvor pametnih delcev (thinking particles standard emitter), ki je močnejši od navadnega oddajnika. Učinkovalec kloniranja sem zvezal na oddajnik, tako da oddajnik odda vsakič en klon učinkovalca kloniranja. Objekti, ki jih oddajam, so navadne krogle z učinkovalcem naključja (random effector), ki jim s pomočjo šuma spreminja obliko. Naključno barvo oddajanih krogel sem najprej hotel doseči z učinkovalcem naključja, ki sem mu način nastavil na barvni kanal. Ustvariti sem moral še material za barvno senčenje (color shader). Z atributom za spreminjanje moči (mix strength) na materialu lahko nato uravnavamo maksimalno deviacijo barve na vsakem objektu, ki nosi ta material. Tega nisem uporabil zato, ker to žal ne dela na klonih oddajnikov. To sem raje rešil tako, da sem ustvaril več instanc materiala in vsaki posebej spremenil barvo. Nato sem nastavil učinkovalcu kloniranja iterativni način, kar pomeni, da iterira od začetka do konca objektov, ki jih mora klonirati, in vsakič izbere naslednjega v vrsti.

5.3 Tretji del videa: oddajanje v smeri zlepka

Tukaj sem hotel prikazati atomski prostor oz. še manjši prostor, v katerem naj bi se nahajale, po teoriji, strune.

Najprej sem naredil zlepek, po katerem sem oddajal 8-kotnike. Oddajnik sem postavil na začetek zlepka, kamero pa na konec. Obema sem dodal skripto poravnava z oznako zlepka (align to spline tag), ki nastavi pozicijo objekta na krivulji na podlagi vrednosti od 0–100, 100 predstavlja končni del krivulje. Različno obliko 8-kotnikov sem dosegel z animiranjem parametrov v samem oddajniku, kot so končna velikost delca, variacija tega parametra, kot, pod katerim oddajnik oddaja, oddajnikova velikost itd. Ker je kamera zarotirana vedno v smeri krivulje, jaz pa sem hotel, da gleda proti krivulji, sem spisal preprosto python skripto, ki pridobi rotacijo kamere, jo zavrti

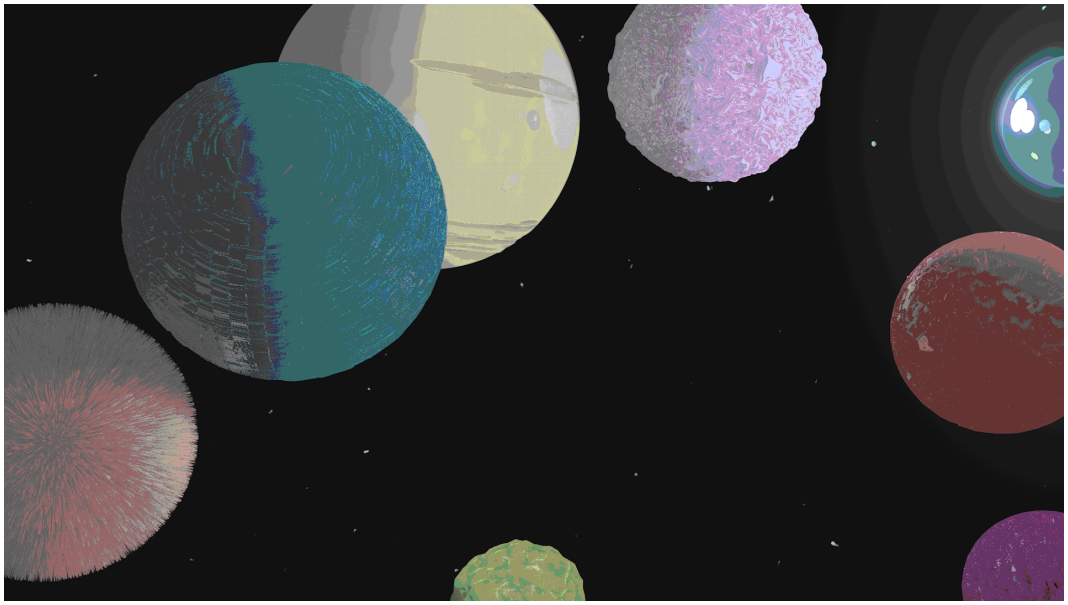


Slika 5.4: 8-kotniki, kateri se premikajo proti kameri s pomočjo zlepka.

in posreduje kameri. To sem naredil z urejevalnikom Xpresso. Ker sem hotel, da se krivulja upogiba, sem naredil krivuljo iz treh točk. Ker se v Cinema4D ne da premikati oz. animirati točke na krivulji, sem uporabil točkovno vozlišče. Kar točkovno vozlišče naredi, je, da vzame točkovni indeks (v mojem primeru 1, ker sem hotel izbrati sredinsko točko) in ga zveže s pozicijo objekta. Objekt, s katerim sem ga zvezal, je ničelni objekt, ki sem ga nato animiral in se je z animirano pozicijo tega objekta spreminjala tudi krivulja. Potem sem prišel do naslednjega problema, saj se ob upogibanju krivulje spremeni dolžina krivulje, in zato objekti, ki potujejo po krivulji, začnejo potovati nazaj po krivulji, saj so vezani na procentualno dolžino krivulje. Tega sem se lotil tako, da sem zamaknil pozicijo oddajnika za toliko, kolikor odstotkov se je povečala dolžina krivulje. S pomočjo python skripte sem si izpisal dolžino krivulje in izračunal, za koliko odstotkov se poveča krivulja. Ko sem izračunal odstotke, sem te odstotke vstavil v odmik pri transformaciji zlepka (offset spline warp), ki je nato zamaknil pozicijo oddajnika.

5.4 Četrti del videa: spreminjanje nastavitev materiala in zvočni učinkovalec

Zadnji del sem si zamislil kot prikaz vesolja. Kamera najprej potuje ven iz atomske ravni in nato potuje iz molekularne ravni nazaj v človeško raven, ki smo jo videli na začetku videa, pozneje se še bolj oddalji in pristane v vesoljni ravni. Tukaj potujemo po osončju in raziskujemo zanimive oblike ter barve planetov.



Slika 5.5: Planeti z različnimi teksturami šuma

Za ozadje sem uporabil objekt neba s teksturo šuma. Objekt neba je neskončno velika krogla, ki ima center na sredini koordinatnega sistema. Vsi planeti so v resnici krogle. Različen videz jim dajejo materiali, ki so za vsak planet drugačni. Poleg različne barve vsakega materiala je glavna stvar, ki posreduje videz planetov, mapiranje prestavitve. Mapiranje prestavitve je črno-bela tekstura, na podlagi katere se 3D-objekt deformira, torej se deformira oblika objekta na podlagi te teksture. Za videz teksture sem uporabil šum oz. različne oblike šuma, ki sem jih raztegoval, deformiral in spreminjal.

Za premikanje šuma skrbi animacijski atribut. Moč deformacije uravnava višinski parameter. Pri nekaterih planetih sem uporabil plasti šumov, ki se nato združijo v en unikaten šum. Vsak planet potuje po svoji elipsi, ki je unikatno velika in rotirana. To omogoča poravnava z oznako zlepka. Za sonce sem uporabil poseben material, ki ima vključene odbojnost, svetilnost ter sij. Pri svetilnosti sem uporabil poseben senčilnik, imenovan barvni senčilnik, katerega namen je, da lahko njegovo barvo spreminjamo s pomočjo drugih učinkovalcev. V mojem primeru barvo spreminja zvočni učinkovalec. Ta, kot sem ga že opisal, spreminja parametre objektov na podlagi frekvenčnega grafa. V mojem primeru sem ta graf omejil samo na nizke frekvence, torej udarce na boben, ter poslal signal v barvni senčilnik. Z zvočnim učinkovalcem sem tudi spreminjal velikost sonca. Zvočni učinkovalec sem nastavil tako, da vzame vedno največjo frekvenco kot vhod. Ker je prišlo do nelinearnega učinka spreminjanja sonca, sem dodal učinkovalec zamude, ki vsako animacijo objekta, na katerega učinkuje, zamakne oz. ublaži. S tem sem dobil bolj gladko spreminjanje Sonca. Sij sem uporabil zato, da dodam objektu neko sevanje. Spremenil sem parametre in označil, da vzame trenutno barvo materiala za barvo sevanja. Luč, ki sem jo uporabil v tej sceni, je tipa omni, kar pomeni, da se ta luč obnaša kot luč v pravem svetu – torej sveti v vse smeri enakomerno. Za dolžino, do katere seže ta luč, sem dal zelo veliko številko, tako da obsije vse objekte v sceni. Luči sem dodal še leče, ki dodajo sceni bolj filmski učinek, kot bi bila posneta s kamero. Luč sem postavil v center Sonca, vendar sem naletel na težavo, da je objekt Sonca blokiral luč. To sem rešil tako, da sem dodal v seznam objektov, ki jih ta luč ignorira, objekt Sonca. Kamero sem animiral na roke.

5.5 Sestavljanje upodobljenih delov videa v programu After Effects

S posebnimi učinki, ki jih lahko videu dodajamo v programu After Effects, nisem pretiraval, pač pa sem ta program uporabljal predvsem za sestavljanje

posameznih delov diplomske naloge, ki so bili upodobljeni v Cinema4D. Želel sem dodati samo nekaj filmskih detajlov, da bi bil video videti bolj realno. Tipično imajo filmi, narejeni z realno kamero, nekaj analognega šuma. To sem dodal s šumom čez celotni video v After Effects.

Za prehode med različnimi dimenzijami sem animiral mapiranje prestavitve po horizontali. Za teksturo te mape sem uporabil navadno črno-belo teksturo z animirano pravokotno masko in njenim alfa kanalom. Ob vsakem prehodu sem povečal šum, da se je videl učinek v čisti temi.

Poglavje 6

Sklepne ugotovitve

Pri izdelavi diplomske naloge sem se seznanil z glasbeno vizualizacijo. Najprej sem predstavil postopke za avtomatsko glasbeno vizualizacijo (avtomatsko transkripcijo glasbe in izgradnjo vizualnega modela), nato sem predstavil različna orodja za prej upodobljeno vizualizacijo, bolj specifično programa Cinema4D ter njegove vtičnike in Adobe After Effects. Na koncu sem še opisal potek izdelave moje vizualizacije, ki je nastala v okviru praktičnega dela diplomske naloge. Ugotovil sem, da je izdelava prej upodobljene 3D-animacije zelo iterativen postopek. Za izdelavo končnega izdelka sem potreboval mnogo več verzij, kot sem pričakoval (veliko njih sem zavrgel). Ugotovil sem tudi, da je upodabljanje dolgotrajen postopek (za eno poglavje moje animacije je računalnik upodabljal več dni). Pomembno je, da natančno planiramo vse elemente animacije vnaprej. Iteracija je v tem primeru hitrejša, in bolj sistematična. Diplomska naloga je dostopna na naslovu [24].

Literatura

[1] (2017) Kvintni krog. Dostopno na: http://www2.arnes.si/~soppciuh/krog2a.png.

[2] (2018) wikipedia - midi. Dostopno na: https://en.wikipedia.org/w/index.php?title=MIDI&oldid=821447220.

[3] (2017). Animation Software: Which One Should You Use? Dostopno na: https://www.bloopanimation.com/animation-software.

[4] (2017). Cinema 4d. Dostopno na: https://en.wikipedia.org/wiki/Cinema_4D.

[5] (2017). Cinema 4d. Dostopno na: http://im.scv.si/wiki/index.php/Cinema_4D.

[6] (2017). Cinema 4d. Dostopno na: http://motionleague.com/wp-content/uploads/2011/03/Screen-shot-2011-03-10-at-11.31.38-PM.png.

[7] (2017). Material editor. Dostopno na: http://midsiku.net/midsiku/adsr_envelope01.gif.

[8] (2017). Maya. Dostopno na: https://i.ytimg.com/vi/l0EsIKgrznw/maxresdefault.jpgg).

[9] (2017). Node editor. Dostopno na: https://i1.creativecow.net/u/64432/monoflopxpresso.png.

[10] (2017). SolidWorks. Dostopno na: https://pro.radeon.com/wp-content/uploads/sites/2/2017/02/Palatov-D2-SOLIDWORKS-RealView-APR-170109-1-1.png).

[11] (2017). VJingr. Dostopno na: http://cdm.link/app/uploads/storiespre2k6/vellolondon.jpg.

[12] (2017). Zbrush. Dostopno na: https://i.ytimg.com/vi/Rxf9JaG-3fo/maxresdefault.jpg).

[13] (2017) Stephan Bodzin. Singularity. Dostopno na: https://www.youtube.com/watch?v=AaxFuXpcQmE.

[14] (2017) Cgees. What to choose — 3Ds MAX VS MAYA VS CINEMA 4D. Dostopno na: https://www.cgees.com/3ds-max-vs-maya-vs-cinema-4d.

[15] Peter Ciuha, Bojan Klemenc, and Franc Solina. Visualization of concurrent tones in music with colours. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1677–1680. ACM, 2010.

[16] Helena Erzetič, Blaž Gabrijelčič. *3D od točke do upodobitve / Blaž Erzetič, Helena Gabrijelčič.* 2009.

[17] Jinyu Han. *Musical Sound Source Separation.* 2015.

[18] Bojan Klemenc. *Vizualni model harmoničnih odnosov med simultanimi toni v glasbi na podlagi psihoakustičnega modela percepcije zvoka.* PhD thesis, Univerza v Ljubljani, 2016.

[19] Arpi Mardirossian and Elaine Chew. Visualizing music: Tonal progressions and distributions. In *ISMIR*, pages 189–194, 2007.

[20] (2017) MatterHackers. Finding the Right 3D Modeling Software For You. Dostopno na: https://www.matterhackers.com/articles/finding-the-right-3d-modeling-software-for-you.

[21] Miha Pešič. Avtomatska transkripcija zvočnih posnetkov tolkal, 2016.

[22] M. Ryynanen, T. Virtanen, J. Paulus, and A. Klapuri. Accompaniment separation and karaoke application based on automatic melody transcription. In *2008 IEEE International Conference on Multimedia and Expo*, pages 1417–1420, June 2008.

[23] Craig Stuart Sapp. Harmonic visualizations of tonal music. In *ICMC*, volume 1, pages 419–422, 2001.

[24] (2017) Julijan Strajnar. Singularity. Dostopno na: `https://goo.gl/RtA3yy`.

[25] Petri Toiviainen. Visualization of tonal content in the symbolic and audio domains. *Computing in Musicology*, 15, 2007.

[26] Peter vergo. *Music of Painting: Music, Modernism and the Visual Arts from the Romantics to John Cag.* 2012.

[27] Emmanuel Vincent. *Blind Audio Source Separation.* Queen mary, 2015.

[28] (2017) Wikipedia. Autodesk 3ds Max. Dostopno na:: `https://en.wikipedia.org/wiki/Autodesk_3ds_Max`.

[29] (2017) Wikipedia. COFFEE (Cinema 4D). Dostopno na: `https://en.wikipedia.org/wiki/COFFEE_(Cinema_4D)`.

[30] (2017) Wikipedia. Short-time Fourier transform. Dostopno na: `https://en.wikipedia.org/wiki/Short-time_Fourier_transform`.

[31] (2017) Wikipedia. Sinesteti. Dostopno na: `https://sl.wikipedia.org/wiki/Sinesteti%C4%8Dnost`.

[32] (2017) Wikipedia. VJing. Dostopno na: `https://en.wikipedia.org/wiki/VJing`.

[33] (2017) Wikipedia. VJing media personality. Dostopno na: `https://en.wikipedia.org/wiki/VJ_(media_personality)`.